REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		Final Report 16 Oct 92 - 30 Sep 95

4. TITLE AND SUBTITLE
A C ++ Formulation for Particle-In-Cell Simulations

5. FUNDING NUMBERS
2301/ES
61102F

6. AUTHOR(S)
Dr N. T. Gladd

AFOSR-TR-95

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Berkeley Research Asssociates
P O Box 241
Berkeley, CA 94701

0787

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NE
110 Duncan Avenue Suite B115
Bolling AFB DC 20332-0001

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
F49620-93-C-0001

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT

The results of this research effort are embodied in a software package called "OOPIC". OOPIC is indeed an efficient, accurate 2.5-D relativistic, electromagnetic PIC code and has the capability of dealing with complex device geometries and a variety of physically relevant boundary conditions. OOPIC also has a comprehensive graphical user interface that facilitates the setup and control of vacuum electron device simulations and provides a variety of scientific visualizations of evolving simulations. Finally, OOPIC possesses an advisor component which serves two purposes -- (1) it mediates interactions between the PHYSICS and GUI subcomponents using it possible to run physics simulations on general C++ platforms without the computer dependent GUI, and (2) it has an embedded rule-based expeert systems that acts to prevent erroneous input by offering advice about input parameters.

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Final Technical Report
A C++ Formulation for Particle-In-Cell Simulations

F49620-93--C-0001

N. T. Gladd
Berkeley Research Associates
PO Box 241
Berkeley, CA
94701

Recapitulation of research objectives:

1. Perform research in collaboration with George Mason University, the University of California at Berkeley, and FM technologies, Inc., with the objective being the formulation of an efficient, accurate 2.5-D relativistic, electromagnetic PIC code written in the C++ language.

2. Formulate and design the basic new code structure in a completely object-oriented manner in accordance with the state-of-the-art in computer science.

3. Include in the research software a practical graphical user interface system which will convey to the using researcher the complete look and feel of a true computer experiment.

4. Document all physical algorithms and computational formalisms employed in all resultant software packages.

Status and overview of research effort:

This research program is a collaborative effort and the project status I describe reflects the effort of all the researchers. After describing the overall status of the project, I will outline particular aspects of the project with which I have been most associated personally.

At the time this report is written, OOPIC is conceptually complete and undergoing beta test. Although the success of the project ultimately depends on public acceptance of the system, the original goals of the project have been met. OOPIC is indeed an efficient, accurate 2.5-D relativistic,

electromagnetic PIC code and has the capability of dealing with complex device geometries and a variety of physically relevant boundary conditions. OOPIC also has a comprehensive graphical user interface that facilitates the setup and control of vacuum electron device simulations and provides a variety of scientific visualizations of evolving simulations. Finally, OOPIC possesses an advisor component which serves two purposes -- 1) it mediates interactions between the PHYSICS and GUI subcomponents making it possible to run physics simulations on general C++ platforms without the computer dependent GUI, and 2) it has an embedded rule-based expert systems that acts to prevent erroneous input by offering advice about input parameters. The basic OOPIC architecture is shown in Figure 1, together with the relationship between its three major subcomponents -- PHYSICS, GUI, ADVISOR. It is important to emphasize that, just as planned, OOPIC is written entirely in C++ and the object-oriented programming paradigm is almost exclusively used. I feel I can speak for the other developers in saying that the decision to develop OOPIC in C++ was well founded and that the progress of the project was enhanced by this choice. In this report, I will concentrate on describing the ADVISOR subcomponent that was my primary responsibility.

Specific efforts and involvements during this year:

In the third and final year of effort I have had significant involvement in the following -

- ***Refinement of the ADVISOR architecture and components:*** My primary work activity this year have centered on refining the design of the ADVISOR and coordinating its operation in conjunction with the PHYSICS and GUI. The architecture of the ADVISOR is discussed in detail below

- ***Generalizing the rule handling capability of the ADVISOR:*** The original capabilities for handling rules constraining parametric input were quite limited. This was because the successful integration of the ADVISOR, GUI, and PHYSICS was paramount even a limited rule handling capability would suffice while addressing issues of integration. This year I have undertaken to generalize and extend the rule handling capability of the advisor. The new facility of rule handling is discussed below.

- **Beta testing activities:** As OOPIC has approached conceptual completeness, our attention has naturally turned to the extensive testing of the system. It was especially important to obtain the opinion of outsiders who were not involved in the detailed development of this system. To this end, I traveled to Philips Laboratory to give a demo of OOPIC and to elicit the opinions of Pat Helles, Moe Armon and their colleagues. A detailed report of this activity is included.

- **Hosting the Spring 95 OOPIC meeting:** Although little technical effort was involved, a surprising amount of administrative effort was expended in arranging for and hosting the Spring 95 OOPIC meeting.

Presentations to learned societies:

- Issues in Providing Expert Advice for Users of a Particle-In-Cell Simulation Code
  N. T. Gladd and J. P Verboncoeur
  Presented in IEEE ICOPS in Madison Wisconsin

Interactions with OOPIC team members

- Approximately weekly meetings with the Plasma Theory and Simulation Group at the University of Cal (John Verboncoeur, Peter Mardahl, Keith Cartwright, Ned Birdsall)

- Spring OOPIC meeting at Walnut Creek

- Conference calls

Architectural structure of the ADVISOR subcomponent:

The advice giving subcomponent of OOPIC consists of a single object of type AdvisorManager that we will call the ADVISOR. The ADVISOR is conceptually very simple -- it is a composite object that holds generic information such as a list of known parameter group types, a list of known boundary types, a list of known probe types, and a string containing information about any errors related to parameter values. This global information is used exclusively by the GUI to set up visual displays and dialogs for the various simulation parameters. The ADVISOR also contains a list of SpatialRegionGroups that hold the parametric information specific to a given simulation. The ADVISOR data structure is illustrated in Figure 2.

A SpatialRegionGroup is a composite object that corresponds to a PHYSICS based SpatialRegion object. Within the PHYSICS, a SpatialRegion object is intended to contain all of the information and techniques necessary to simulate a given "spatial region" of a model device. This concept is discussed in detail in Verboncoeur, Langdon and Gladd [Computer Physics Communications]. A SpatialRegion object is a composite object containing several conceptual clusters of parameters such as those required to specify a computational grid, or those corresponding to control of simulation algorithms. An ADVISOR based SpatialRegionGroup is a database object that contains the parametric information necessary to create a PHYSICS based SpatialRegion. It's data elements exactly mirrors the data elements that parameterize the PHYSICS based SpatialRegion, i.e., there is a ParameterGroup object corresponding to each named conceptual cluster of parameters, and, within each ParameterGroup, there is a Parameter object corresponding to each named parameter within a conceptual cluster. This mirror image relationship of the ADVISOR's SpatialRegionGroups and the PHYSIC's SpatialRegions is useful and illustrates the power of object oriented programming concept of polymorphism. When it is decided that a simulation should be initiated, the single message "CreateCounterPart" is sent to the ADVISOR which passes this message to each of its various component objects, which responds by creating a PHYSICS object corresponding to itself. The mirror symmetry of the ADVISOR and the PHYSICS is illustrated in Figure 3. The SpatialRegionGroup [shadowed box] creates a counterpart of itself, each of the contained ParameterGroups [rectangles] creates its counterpart, and, in turn, each of the contained Parameters

[shapes] creates its counterpart.

.....
And the big fleas have little fleas upon 'em,
On and on, Ad Infinitem.

The ADVISOR also performs database functions for the GUI. The GUI is capable of accessing the data contained within an ADVISOR's SpatialRegionGroup and displaying it so that it can be viewed and modified by users. The general operating procedure is that the GUI initiates the ADVISOR by passing it the name of a file. The ADVISOR reads the input file and parses it into ParameterGroup and Parameter objects and makes these objects available to the GUI for modification. As we will discuss below, the ADVISOR also has the ability to decide whether changes made by the GUI are "legal". Typically, after several iterations between the GUI and ADVISOR, a user decides that the parameters are satisfactory for the simulation in mind. At that time the GUI instructs the ADVISOR to create the requisite PHYSICS objects and the simulation begins. At that time, the GUI can make additional changes to parameters in anticipation of a future simulation.

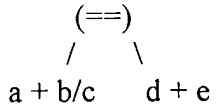
Rule handling capability of the ADVISOR: One of the important functions of the advisor is to detect the choice of inappropriate input on the part of users and warn them of necessary or required changes. This is the primary way in which "advice" is provided to OOPIC users. As currently implemented, designers or users can provide rules that constrain the relationship between any parameters within a conceptual cluster of parameters, i.e., within an ADVISOR ParameterGroup. For example, one could warn of a fatal error in setting up the computational grid with a rule like [J < 2, Fatal! Illegal grid] that would notify the user that J, the number of grid points in one direction, was too small to constitute a legal grid and that the user specified value would cause a fatal error in the code. Alternatively, a rule like [J*K > 10000, Warning! Too many grid points will result in slow operation] could warn users that their grid may be too finely resolved to permit reasonable turnaround times for calculations.

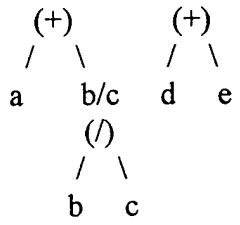
Originally, the rule handling capability of the advisor was limited to a few simple forms such as those given above. This year the rule handling capability has been substantially enhanced to allow almost any combination of algebraic and logical expressions. This has been accomplished by translating an algebraic/logical expression representing a rule into a tree structure and then evaluating the tree in a recursive manner.

This is best illustrated by an example. Consider the expression

$$a + b/c == d + e$$

This is recursively parsed into the tree





The expansion of the tree continues until the leaves are either a symbol or a numerical value. To evaluate the tree, all symbols are substituted for their numerical values and then the tree is expanded. First the operation == is attempted. It is found that the leaves of the top tree are expressions so the value of the next level leaves are attempted. This continues until a subtree is found that can be explicitly evaluated is found, i.e., it consists of an operator node and numerical leaves. The evaluation then passes back up the tree structure until a final numerical value for the overall tree is determined.

It should be noted that it is very important that the operators be chosen in a particular order when parsing the expression into a tree. Specifically, one must parse first on logical operators such as && and ||. The various comparison operators, ==, >=, >, etc., are then parsed. After that, the algebraic operators + and - must be parsed before the operators *, /, and ^.

There are other details to consider. For example, parentheses are handled by temporarily tokenizing parenthetical subexpressions during the expansion procedures. Another complication results from the fact that the symbols "+" and "-" may appear in scientific notation numbers as well as operators in the overall expression. These details are handled by improving the pattern matching capabilities of the expression parser.

In conclusion, this new rule handling capability will greatly increase the complexity of rules that can be specified while simultaneously simplifying the ADVISOR code, e.g., there is one type of rule rather than three.

Report on Philips beta test:

On February 10, I visited the Philips Laboratory and delivered the initial beta release of OOPIC. The executable released was betaexe.zip and the associated source code was b02-07.zip - both obtained on February 8 from Hickory. As I will comment later, I do not think that betaexe.zip actually derives from the b02-07.zip source.

I had an excellent visit at Philips and was treated with great hospitality by the Philips personnel. I met with Pat Helles, who will be the primary beta tester, and also with Moe Armon, Les Bowers, Tom Hussey and Leon Chandler. These people offered many helpful suggestions that I will detail below.

My original plan was to install only the betaexe.zip executable on Pat's P90 and to observe his use of the program - offering only minimal comments and/or help. However, Moe and Les were already

at the office when I arrived and would only be available for only an hour or so. Therefore, I gave a demo of the version of OOPIC I had on my portable (compiled from b02-07.zip). In retrospect, this went very well and there was no shortage of useful suggestions.

In what follows I will list the various comments and suggestions that were made, separating them into two categories - 1) physics and modeling, and 2) GUI and aesthetics. I will paraphrase from the notes I kept and try to indicate who made each comment.

1) Physics and modeling:

I began by showing the default case of the beam in a tincan. Moe immediately began asking questions about the physical parameters characterizing the beam. The default case has a low current (I = 0.01 amp), non-relativistic beam (v_{z0} = 1.0e8 m/sec). Moe suggested we increase the current to the physically interesting value of 50kAmp and observe whether a virtual cathode would form. I made that change and the system hung. After some discussion, we realized that the number of physical particles per simulation particles should have been increased so that the much higher current would not give rise to a huge number of simulation particles. We made a couple of incremental increases of the current and number of simulation particles and were able to achieve a high current beam that definitely showed the formation of a virtual cathode. The system hang was almost certainly associated with OOPIC trying to create a very large number of simulation particles. Since this is a problem that can occur in many different circumstances I will try to formulate some "rules" that will warn users if the indicated number of simulation particles will be very large.

Moe asked questions about the physical assumptions underlying the beam emitter and I was unsure of the answers. We should take the time to write clear descriptions of the physics basis and assumptions of the various boundary elements.

Moe asked whether we plotted momentum instead of velocity for relativistic cases -- we don't. His point is well made and we should consider how to plot the momentum. At the least, we should make it an option.

Currently, we specify beam characteristics in terms of velocities. Specification of beams in terms of energy would be more appropriate for most simulations of physical significance. How about the idea of having double entry boxes for velocities - one for normal velocity input, one for energy? When an entry was made for energy, the velocity value would become self-consistent - and vice versa.

Moe made the point that our default and example cases should correspond to high voltage microwave situations that are of interest to users. He felt that it is important to demonstrate the nonlinear phenomena, e.g., virtual cathode formation, that is important to simulators and system designers. I saved the parameters associated with Moe's virtual cathode example and will make that the standard default case.

I demonstrated the klystron example in which we use a gap exciter to modulate a hollow beam. Moe

wondered why we "cheated" and did not include a real cavity - a situation more closely related to some of his current work. Using the geometry editor, I created an outer boundary that resembled the cavity shape that Moe suggested and saved the parameters associated with it. I will talk with him further and try to set up an example case that models his experiment.

Les commented on the ease of constructing the shaped outer boundary but raised the issue of the labor involved when many such structures are required. He currently uses MAGIC's DO LOOP capabilities to construct such boundaries. We talked about the possibility of marking a shape consisting of several boundary elements in the geometry editor and using it as a "cookie cutter" to stamp out repetitive cases of that shape. We should think about how to do this since it guaranteed to occur in the simulation of realistic devices.

Leon raised the important issue of restart capabilities. We need to address this soon!

2) GUI and aesthetics:

Pat Helles suggested that the labeling of the RUN button should switch to STOP when the program was running.

He also noted that although the system comes up in the Geometry-Editor mode, the corresponding tab on the left hand side is not blanked. This is an initialization problem since it works properly after a switch has been made to one of the other windows and back.

Pat noticed that when the Geometry-Editor is expanded to full-screen, the various axis labeling can overlap the plot region. This can probably be fixed with some logic based on the size of the exposed window.

When we looked at dialog boxes on Pat's machine (with a very large monitor), the numerical entries were truncated and almost unreadable. Pat eventually found that this could be fixed by changing the font size setting on his video driver.

Pat, who tried all sorts of things I would never think about, found that once he closed a window on the initial condition screen, it could not be recovered. This is bad because it means the parameters associated with the closed window cannot be changed. I can think of no reason why these windows should ever be closed so we should disable the closing feature for those windows.

Pat is a keyboard maestro (like John V) and disliked having to mouse from box to box to change dialog values. Standard WINDOWS practice would cycle us through boxes with TAB and SHIFT TAB. We should implement this feature.

Both Pat and Leon noticed that when there is an error in a dialog box, the only sign is that the top of the box turns magenta. It was felt that the top of the box should print out "There are errors" and/or beep. Pat would go even further and play an error.wav file that would state that errors had

occurred. They also felt that the errors should be listed at the top of the dialog box, rather than at the bottom as they currently are listed. Leon thought it would be nice if the particular parameters for which errors had occurred appeared in the dialog box with red color coding.

Pat thought that the message that appears before running a case was ambiguous. The pop up dialog box should say "You have made changes to the simulation setup. Do you want to impose those changes in the physics before running?" It is not clear to users that changes made in the geometry editor are not immediately reflected in the physics. We need to find some way to clarify this situation.

Both Pat and Leon thought that the standard hourglass icon should appear when a mouse message gesture is made and the system is unable to respond immediately.

While running examples, Pat and I noticed

- printing a particle plot resulted in a plot without any labeling or axes. I was surprised since I remembered that this was working at one time.
- history plots did not respond to a print command.
- attempting to plot a greyed box on the history menu hangs the code.
- there were inconsistencies between the E-field menu boxes and the plots that were in view, e.g., killing an E-field window did not remove the check in the menu box.
- the auto-scaling of the 3d plots does not work. In cases where a periodic field is being applied to the system, this makes the 3d windows very erratic and difficult to understand.
- The viewpoint window doesn't seem to do anything. Furthermore, it is not clear how to get rid of it. Playing around showed that it could be toggled off. Why doesn't it have a close capability?
- Three D windows should have a reset button that would take them back to their original condition.
- We were unable to successfully add probes and produce plots associated with them.

Leon suggested that the dialog boxes should provide more information about the parameters. For example, he suggested that mousing on a parameter name should bring up a definition of the parameters and an indication of the physical units with which it is specified.

Leon also thought that we should make more use of HELP keys to provide information about the

screens being viewed.

Things that I noticed:

- The beta executable was not consistent with the beta source! The proof of this is that the Control Parameters screen does not appear in the executable (thus the magnetic field could not be changed). The Control Parameters screen is available when b02-07.zip is compiled.

- The executable generated by Mark does not work on all machines. It didn't work on any of my three machines (486 desktop, portable and P90) but did work on Pat's machine (P90). We need to determine how to deliver an executable that will work on all machines. This may require static binding of all libraries.

- There are a number of buttons, options, choices, etc. in the GUI that do not correspond to working options. These can cause a lot of confusion and should be removed from the user's view until we are sure they are working and should be removed from the user's

- Occasionally, when you try to close a window, the image is shifted to the upper left hand of the screen and does not go away. This eventually makes the simulation unusable and you have to restart.

- The system crashed several times during the demo I gave for reasons that are not clear to me. This will not be acceptable in a 1.0 release so we have to track these crashes down.

- Related to the modeling of realistic devices is the issue of showing geometries with many grid points. The limited number of pixels on the screen makes viewing and manipulating grid sizes of more than 100x100 very unwieldy. We need to be able to show portions on the device in a scrolling window but also have the possibility of showing an overview of the whole device.

- It is crucially important to stabilize the operation of the status bar so that we can track how a simulation is proceeding.

Summary: This trip to Philips was very useful! I cannot stress enough how helpful and hospitable Pat Helles and his colleagues were. Pat, who arranged it so that I was exposed to both Mexican food and chicken-fried steak, certainly earned the appreciation of this displaced Texan. We will look forward to hearing comments from Pat and Moe at our meeting next month.

From the list of things appearing above, it is clear that we have a LOT TO DO. My personal feeling is that we should freeze new code development until we have fixed the majority of the problems that are pointed out by the Philips, Maryland and Stanford beta testers.

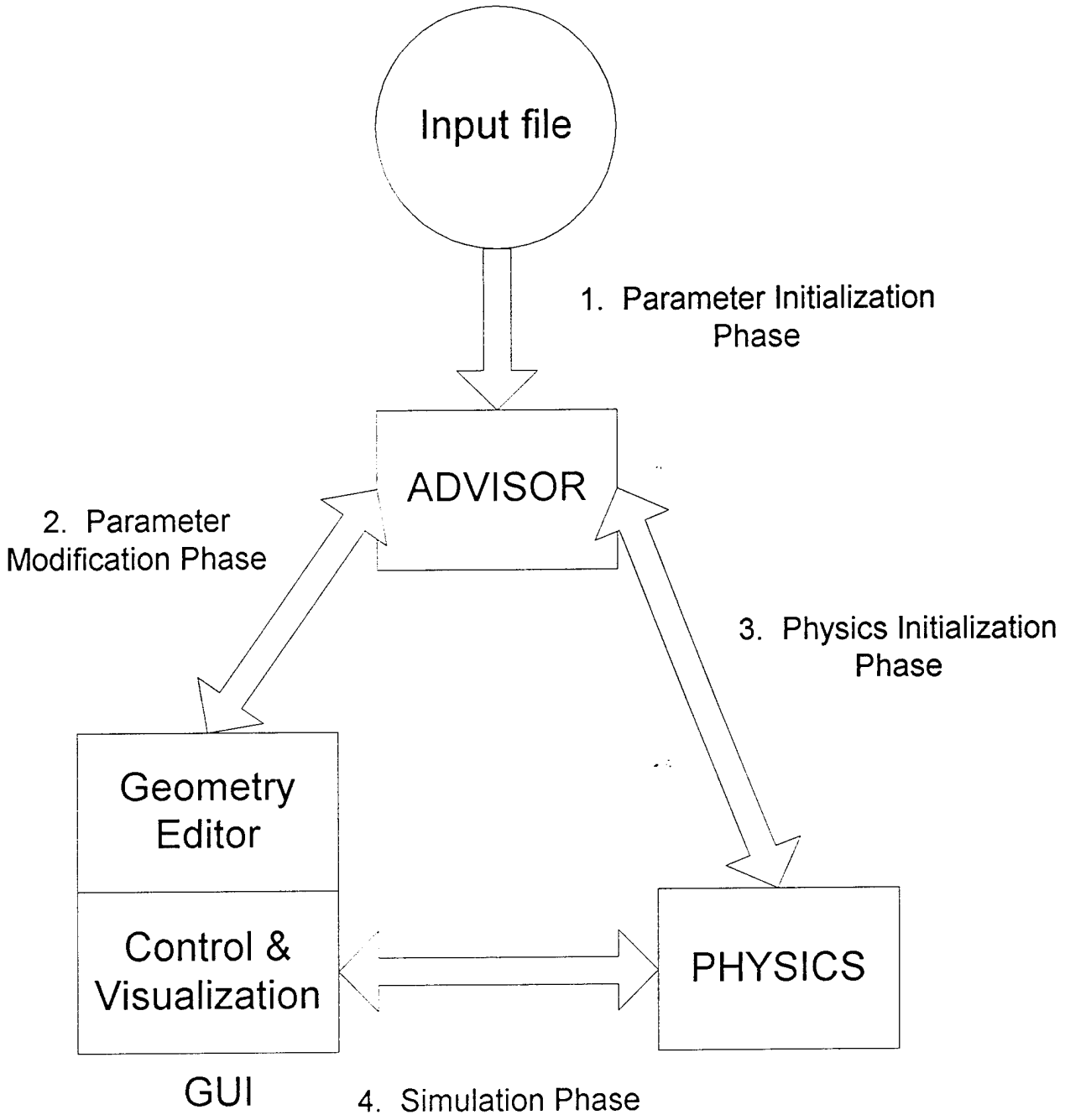


Figure 1. Basics components of OOPIC architecture and the important phases of information flow

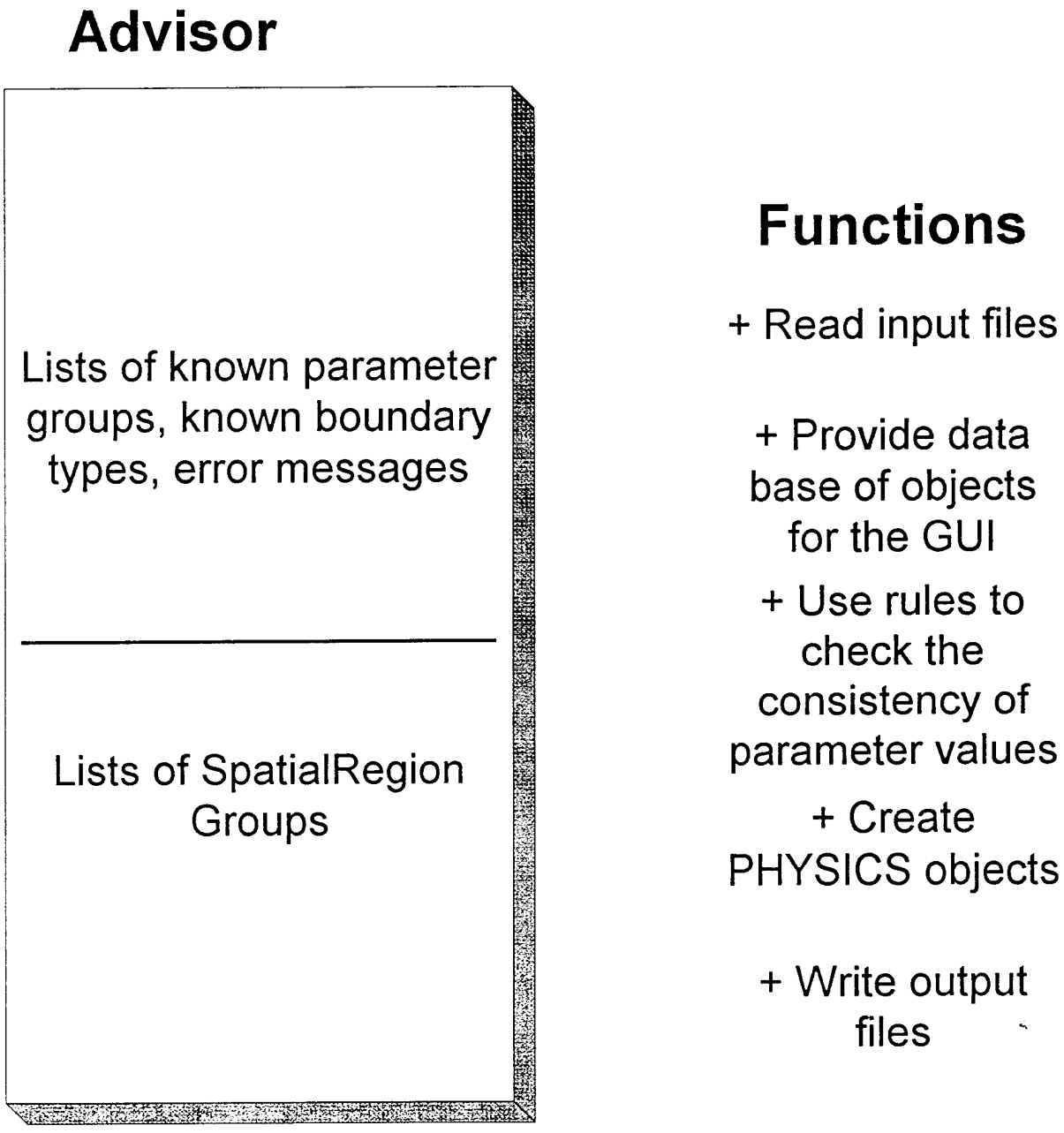


Figure 2 Advisor structure and functionality

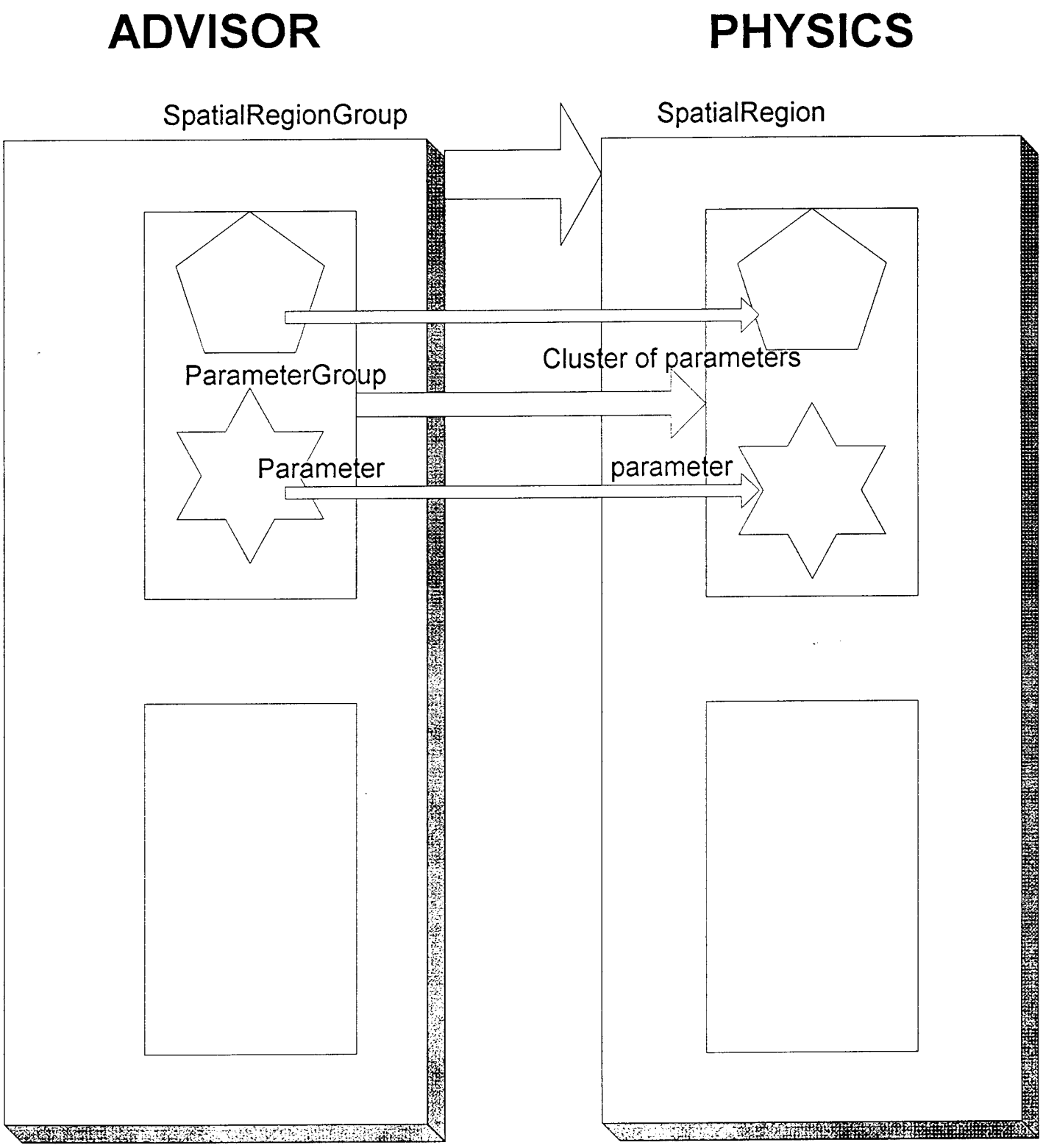


Figure 3 Mirror image of ADVISOR and PHYSICS data objects